A Unified View of Data-Intensive Flows in Business Intelligence Systems: A Survey

Petar Jovanovic, Oscar Romero, and Alberto Abelló

Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain ({petar|oromero|aabello}@essi.upc.edu)

Abstract. Data-intensive flows are central processes in today's business intelligence (BI) systems, deploying different technologies to deliver data, from a multitude of data sources, in user-preferred and analysis-ready formats. To meet complex requirements of next generation BI systems, we often need an effective combination of the traditionally batched extract-transform-load (ETL) processes that populate a data warehouse (DW) from integrated data sources, and more real-time and operational data flows that integrate source data at runtime. Both academia and industry thus must have a clear understanding of the foundations of data-intensive flows and the challenges of moving towards next generation BI environments. In this paper we present a survey of today's research on data-intensive flows and the related fundamental fields of database theory. The study is based on a proposed set of dimensions describing the important challenges of data-intensive flows in the next generation BI setting. As a result of this survey, we envision an architecture of a system for managing the lifecycle of data-intensive flows. The results further provide a comprehensive understanding of data-intensive flows, recognizing challenges that still are to be addressed, and how the current solutions can be applied for addressing these challenges.

Keywords: Business Intelligence · data-intensive flows · workflow management · data warehousing

1 Introduction

Data-intensive flows are critical processes in today's business intelligence (BI) applications with the common goal of delivering the data in user-preferred and analysis-ready formats, from a multitude of data sources. In general, a data-intensive flow starts by extracting data from individual data sources, cleans and conforms extracted data to satisfy certain quality standards and business requirements, and finally brings the data to end users.

In practice, the most prominent solution for the integration and storage of heterogeneous data, thoroughly studied in the past twenty years, is data warehousing (DW). A DW system assumes a unified database, modeled to support analytical needs of business users. Traditionally, the back stage of a DW system comprises a data-intensive flow known as the extract-transform-load (ETL) process responsible of orchestrating the flow of data from data sources towards a

DW. ETL is typically a batch process, scheduled to periodically (e.g., monthly, daily, or hourly) load the target data stores with fresh source data. In such a scenario, limited number of business users (i.e., executives and managers) are expected to query and analyze the data loaded in the latest run of an ETL process, for making strategic and often long-term decisions.

However, highly dynamic enterprise environments have introduced some important challenges into the traditional DW scenario.

- Up-to-date information is needed in near real-time (i.e., right-time [39]) to make prompt and accurate decisions.
- Systems must provide the platform for efficiently combining in-house data with various external data sources to enable context-aware analysis.
- Systems must be able to efficiently support new, unanticipated needs of broader set of business users at runtime (i.e., on-the-fly).

These challenges have induced an important shift from traditional business intelligence (BI) systems and opened a new direction of research and practices. The *next generation BI* setting goes by various names: *operational BI* (e.g., [15,22]), *live BI* (e.g., [17]), *collaborative BI* (e.g., [6]), *self-service BI* (e.g., [1]), *situational BI* (e.g., [63]). While these works look at the problem from different perspectives, in general, they all aim at enabling the broader spectrum of business users to access a plethora of heterogeneous sources (not all being under the control of the user's organization and known in advance), and to extract, transform and combine these data, in order to make right-time decisions. Consequently, here, we generalize these settings and use the common term *next generation BI*, while for the old (DW-based) BI setting we use the term *traditional BI*. An interesting characterization of the *next generation BI* setting is given by Eckerson [22]: "*...operational BI requires a just-in-time information delivery system that provides the right information to the right people at the right time so they can make a positive impact on business outcomes.*"

Obviously, in such a scenario periodically scheduled batch loadings from preselected data stores have become unrealistic, since fresh data are required in near real-time for different business users, whose information needs may not be known in advance. In fact, effectively integrating the traditional, batched decision making processes, with on-the-fly data-intensive flows in *next generation BI* systems is discussed to be important to satisfy analytic needs of today's business users (e.g., [2,17,40]). For example, a part of in-house company's sales data, periodically loaded to a DW by an ETL, can be at runtime crossed with the Web data to make context-aware analysis of business decisions. To build such demanding systems, one must first have a clear understanding of the foundation of different data-intensive flow scenarios and the challenges they bring.

From a theoretical perspective, handling data heterogeneity has been separately studied in two different settings, namely *data-integration* and *data exchange*. Data integration has studied the problem of providing a user with a unified virtual view over data in terms of a global schema [59]. User queries over the global schema, are then answered by reformulating them on-the-fly in terms

of data sources. On the other side, data exchange has studied the problem of materializing an instance of data at the target that reflects the source data as accurately as possible and can be queried by the user, without going back to the original data sources [27].

A recent survey of ETL technologies [96] has pointed out that the data exchange problem is conceptually close to what we traditionally assume by an ETL process. Intuitively, in an ETL process, we also create an instance at the target, by means of more complex data transformations (e.g., *aggregation*, *filtering*, *format conversions*, *deduplication*). However, the trends of moving towards the *next generation BI* settings have brought back some challenges initially studied in the field of data integration, i.e., requiring that the user queries should be answered by extracting and integrating source data at runtime. Moreover, the *next generation BI* settings brought additional challenges into the field of data-intensive flows (e.g., low data latency, context-awareness).

Right-time decision making processes demand close to zero latency for data-intensive flows. Hence the automated optimization of these complex flows is a must [17,44], not only for performance, but also for other quality metrics, like fault-tolerance, recoverability, etc. [84]. Considering the increasing complexity of data transformations (e.g., machine learning, natural language processing) and the variety of possible execution engines, the optimization of data-intensive flows is one of the major challenges for *next generation BI* systems.

Even though the above fields have been studied individually in the past, the literature still lacks a unified view of data-intensive flows. In this paper, we aim at studying the characteristics of data-intensive flows in *next generation BI* systems. We focus on analyzing the main challenges in the three main stages when executing data-intensive flows, i.e., (1) *data extraction*, (2) *data transformation*, and (3) *data delivery*. Having low data latency as an important requirement of data-intensive flows in the *next generation BI* setting, we additionally analyze the main aspects of *data flow optimization*. We analyzed these four areas inside the two main scenarios for data-intensive flows: (a) periodically executed, batched processes that materialize and load data at the target data store for future analysis (*extract-transform-load - ETL*), and (b) on-the-fly, instantaneous data flows executed on demand upon end-users' query (*extract-transform-operate - ETO*).

In addition, we identify that a well-known BigData challenge, namely, one of the so called 3 V's [45] (i.e., massive *volumes* of data), is an important one also for the design and even more deployment of data-intensive flows in next-generation BI systems. However, the approaches that deal with such challenge represent a separate and rather extensive field of research, which is out of scope of this study. We thus refer an interested reader to [13] for more detailed overview of the approaches dealing with the BigData challenges.

As the first result, we identify the main characteristics of data-intensive flows, focusing on those that best describe the challenges of moving towards the *next generation BI* setting. Then, in terms of these characteristics we classify the approaches, both from the foundational works and more recent literature, tackling

these characteristics at different levels. On the one hand, the results provide us with a clear understanding of the foundations of data-intensive flows, while on the other hand, identified characteristics help us defining the main challenges of moving towards the *next generation BI* setting.

Finally, as the main outcome of this study, we outlined the envisioned architecture of *next generation BI* systems, focusing on managing the complete lifecycle of data-intensive flows.

Contributions. In particular, our main contributions are as follows.

- We analyzed current approaches, scrutinizing the main aspects of data-intensive flows in today's BI environments.
- We define the main characteristics of data-intensive flows, focusing on those that best describe the challenges of a shift towards the *next generation BI*.
- In terms of the dimensions defined from these characteristics, we analyze both the foundational work of database theory, and recent approaches for data-intensive flows, at different levels of these dimensions.
- Resulting from this study, we envision an architecture for managing the complexity of data-intensive flows in the *next generation BI* setting.
- Finally, we indicate the remaining challenges for data-intensive flows, which require further attention from both academia and industry.

Outline. In Section 2, we first introduce an example scenario used to support our discussions throughout this paper. We then in Section 3, describe the methodology used in our study, and outline the main study setting. Next, in Section 4 we discuss the process of defining the dimensions that are further used for studying data-intensive flows. In Sections 5 - 8, we analyze different approaches from data-intensive flows inside the previously defined dimensions. In Section 9 we provide the overall discussion and introduce an envisioned architecture of a system for managing data-intensive flows in the *next generation BI* setting, while in Section 10, we conclude the paper.

2 Example Scenario

We first introduce an example scenario to support discussions throughout this paper and to motivate our study. Our example scenario is motivated by the data model introduced for the big data benchmark (a.k.a. BigBench) in [33], which extends the TPC-DS benchmark[1] for the context of big data analytics. Notice that we adapted their scenario to make the examples more intuitive and suitable to our discussions. In particular, besides the typical operations found in relational database systems, i.e., *Join*, Selection (*Filter*), Aggregation (*Aggr.*), *Sort*, and Distinct (*Remove Duplicates*), in the following example scenarios, we also introduce more complex operations typically found in today's data-intensive flows; that is, (1) User Defined Functions (*UDF*) that may implement either simple arithmetic expressions or complex, typically black-box operations, (2) *Match*

[1] `http://www.tpc.org/tpcds/spec/tpcds_1.1.0.pdf` (last accessed 4/4/2014)

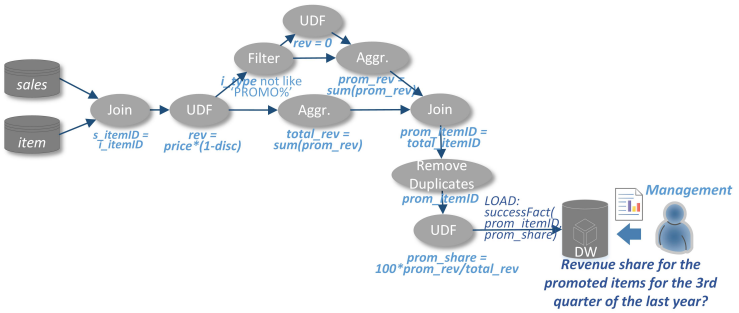

Fig. 1. Example 1.1: ETL to analyze revenue share from a promotion

that implements more relaxed join or lookup semantics (e.g., using approximate string matching), and (3) *Sentiment Analysis* that typically applies natural language processing techniques for extracting subjective (opinion) information from the Web sources (e.g., forums, social networks).

In general, we consider a simple case of a retail company that has different databases that support its daily operational processes. These databases cover the information about different items (i.e., products) offered for the sale, company's customers, their orders, shipping information, etc. Periodically, the company launches campaigns and puts some product on a promotion.

Scenario 1. Originally, the company has used a *traditional BI* system with a centralized DW that is loaded by means of different ETL flows (one of which is conceptually depicted in Figure 1). Users in this scenario are typically upper management executives that analyze the enterprise-wide data (e.g., `items` and their `sales`) to make decisions for making strategic actions (e.g., launching promotional campaigns).

Example 1.1. In the specific example in Figure 1, the ETL flow is periodically executed to load the DW with information about the percentage of the revenue share made from the items that were on the promotion. Quarterly, the management analyzes how the previous business decisions on promotional campaigns affected the revenue.

Scenario 2. While the above setting has served the company well in having a periodical feedback about the previous strategic decisions, today's dynamic markets require more prompt reaction to the potentially occurring problems (e.g., hourly or daily). The company thus noticed that instead of waiting for the sales data to analyze the success of the promotional campaign, they can potentially benefit from the opinions that customer may leave about the campaign and product items, in the form of `reviews` over the Web (e.g., social networks) and react faster to improve the potential revenue. Moreover, the company also noticed that such an analysis should be decentralized to the regional and local representatives and available to a broader set of users involved in the business process. As fast decisions are needed, the users must be able to make them at right time (i.e., "*...before a problem escalates into a crisis or a fleeting opportunity disappears...*"

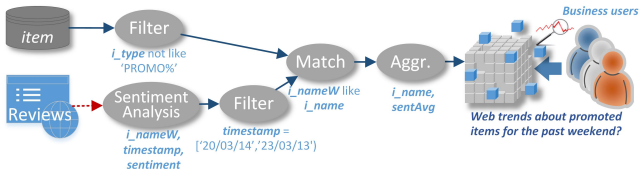

Fig. 2. Example 2.1: ETO to predict the success of a promotion

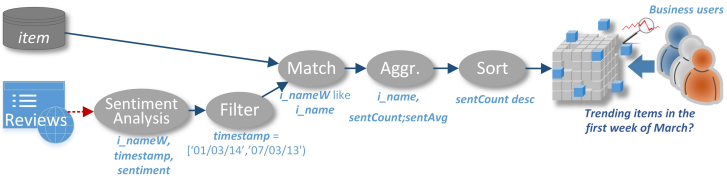

Fig. 3. Example 2.2: ETO to analyze the trends for launching a new promotion

[22]). We consider the two following business requirements posed on-the-fly, and the two data-intensive flows that answer them, conceptually depicted in Figure 2 and Figure 3.

Example 2.1. In the former example (Figure 2), a regional manager decides to analyze the potential success of the promotional campaign launched on Friday, after the first weekend, by inspecting the sentiment (i.e., opinions) of the real and potential customers about the product `items` that are included in the promotion.

Example 2.2. The latter example (Figure 3), on the other hand, analyzes the currently trending product `items` and user opinions about them for deciding which items to include in the next promotional campaign.

In both cases, business users interactively make on-the-fly and context-aware analysis, in order to quickly react and improve their business decisions.

3 Methodology

We further introduce the methodology used for studying data-intensive flows and outline the resulting study setting. We start by introducing the process of selecting the literature to be included in this study. The study further includes three consecutive phases, which we explain in more detail in the following subsections.

3.1 Selection process

The literature exploration started with the keyword search of the relevant works inside the popular research databases (i.e., Scopus[2] and Google Scholar[3]). In

[2] https://www.scopus.com/
[3] https://scholar.google.com

Phase I, we focused on the keywords for finding seminal works in the DW and ETL field (i.e., "data warehousing", "ETL", "business intelligence"), as well as the most relevant works on the *next generation BI* (i.e., "next generation business intelligence", "BI 2.0", "data-intensive flows", "operational business intelligence"). While in the case of traditional DW and ETL approaches we encountered and targeted the most influential books from the field (e.g., [53,46]) and some extensive surveys (e.g., [96]), in the case of the next generation BI approaches, we mostly selected surveys or visionary papers on the topic (e.g., [1,2,17,15,40]). Furthermore, the following phases included keyword search based on the terminology found in the created study outline (see Figure 4), as well as the identified dimensions (see Figure 5).

Rather than being extensive in covering all approaches, we used the following criteria for prioritizing and selecting a representative initial set of approaches that we studied.

- The *relevance* of the works to the field of data-intensive flows and the related topics, i.e., based on the abstract/preface content we discarded the works that did not cover the topics of interest.
- The *importance* of the works, i.e., number of citations, importance of the venue (e.g., ranking of the conference[4] or impact factor of the journal[5]).
- The *maturity* of the works, i.e., extensiveness and completeness of a theoretical study or a survey, experimental results, applicability in the real world.

Furthermore, we also followed the snowballing technique and included, previously not found, but relevant approaches referenced from the initial ones.

3.2 Phase I (Outlining the study setting).

First phase included the review of the seminal works on traditional data-intensive flows, ETL, and data warehousing in general; as well as the relevant works discussing the *next generation BI* systems and their main challenges on moving toward (near) real-time data analysis.

As a result, we outline the setting for studying data-intensive flows. Specifically, in our study we aim at analyzing two main *scenarios* of data-intensive flows present in today's BI settings, namely:

- *extract-transform-load* (*ETL*). In the *traditional BI* setting, data are extracted from the sources, transformed and loaded to a target data store (i.e., a DW). For posing analytical queries (e.g., OLAP), the business users in such a scenario solely rely on the data transferred in periodically scheduled time intervals, when the source systems are idle (e.g., at night time).
- *extract-transform-operate* (*ETO*). *Next generation BI* has emerged as a necessity of companies for combining more instantaneous decision making with traditional, batched processes. In such a scenario, a user query, at runtime,

[4] CORE conference ranking: http://portal.core.edu.au/conf-ranks/
[5] Thomas Reuters Impact Factor: http://wokinfo.com/essays/impact-factor/

gives rise to a data flow that accesses the data sources and alternatively crosses them with already loaded data to deliver an answer.

Kimball and Caserta [53] introduced the following definition of an ETL process *"A properly designed ETL system extracts data from the source systems, enforces data quality and consistency standards, conforms data so that separate sources can be used together, and finally delivers data in a presentation-ready format so that application developers can build applications and end users can make decisions."*

Being general enough to cover the setting of data-intensive flows studied in this paper (i.e., both previous scenarios), we follow this definition and first divide our study setting into three main stages, namely:

i *Data extraction.* A data-intensive flow starts by individually accessing various (often heterogeneous) data sources, collecting and preparing data (by means of structuring) for further processing.
ii *Data transformation.* Next, the main stage of a data-intensive flow transforms the extracted data by means of cleaning it for achieving different consistency and quality standards, conforming and combining data that come from different sources.
iii *Data delivery.* This stage is responsible for ensuring that the data, extracted from the sources, transformed and integrated are being delivered to the end user in a format that meets her analytical needs.

Related fields. To complete the data-intensive flows lifecycle, in addition to the main three stages, we revisit two fields closely related to data-intensive flows, i.e., *data flow optimization* and *querying*.

iv *Data flow optimization* considers data-intensive flow holistically and studies the problem of modifying the given data flow, with the goal of satisfying certain non-functional requirements (e.g., performance, recoverability, reliability). Obviously, the optimization problem is critical for data flows in today's BI systems, where the data delivery is often required in the near real-time manner.

In addition, for the completeness of the overall picture, we briefly analyze what challenges the two main scenarios in data-intensive flows (i.e., *ETL* and *ETO*) bring to *querying*.

v The *requirements elicitation and analysis* (i.e., *querying*) stage mainly serves for posing analytical needs of end users over the available data. This stage is not actually part of a data-intensive flow execution, but depending on the scenario (i.e., either *ETO* or *ETL*), can respectively come as a preceding or subsequent stage for a data-intensive flow execution. At the lower level of abstraction, end users' analytical needs are typically expressed in terms of queries (e.g., SQL), programs (e.g., ETL/MapReduce jobs), or scripts (e.g., Pig Scripts), which are then automatically translated to data-intensive flows

that retrieve the needed data. The typical challenges of *querying* the data in the *next generation BI* setting concern the ability of the system to adapt and complement users' analytical needs by means of discovering related, external data, and the usability of a BI system for posing analytical needs by end-users. The former challenge may span from the traditional DW systems that typically answer user's OLAP queries solely by exploiting the data previously loaded into a DW (by means of an ETL process), to situation-(context-)aware approaches that considering end user queries, explore, discover, acquire, and integrate external data [1,2]. Regarding the latter challenge, we can also observe two extreme cases: traditional querying by means of standard, typically declarative query languages (e.g., SQL, MDX), and approaches that enable users to express their (often incomplete) analytical needs in a more natural and human-preferred manner (e.g., keyword search, natural language). Recently, some researchers have proposed more flexible (semi-structured) query language (SQL++) for querying a variety of both relational and new NoSQL databases that may store data in a variety of formats, like JSON or XML [69].

Other approaches also tackled the problem of providing a higher level of abstraction for posing information requirements, more suitable for business users. As analyzing information requirements and systematically incorporating them into a design of data-intensive flows has been typically overlooked in practice, initial efforts were mostly toward systematic requirements analysis in BI and DW projects (e.g., [36,102]). However, such approaches still require long processes of collecting the requirements at different levels of abstraction and their analysis, before manually incorporating them into a data-intensive flow design. Thus, they obviously cannot be applied in *ETO* scenarios, where the generation of data-intensive flows to fulfill end user analytical needs is expected in near real-time. Other works identified such problem and proposed certain automation to such time-lasting process (e.g., [79]). However, the automation required lowering the level of abstraction for defining information requirements, tightening them to the multidimensional model notation (i.e., facts and dimensions of analysis). As an extension to this approach the authors further tried to raise the level and provide an abstraction of the data sources' semantics in terms of a domain ontology, with its graph representation. This has partially hidden the model specific details from business users, and allowed them to pose information requirements using a domain vocabulary. Recent work in [31] conducted an extensive study of decision support system approaches, identifying how they fit the traditional requirements engineering framework (i.e., [72]). As a result, the authors identified a need for systematic and structured requirements analysis process for further raising the level of automation for the design of decision support systems (including the design of data-intensive flows), while at the same time keeping the requirements analysis aware of all stakeholder needs. We have discussed here the main challenges that *requirements elicitation and analysis* introduces to the field of data-intensive flows in the *next generation BI* setting, but we omit further analysis as it falls out of the scope of this work.

As a result, we define a blueprint for studying data-intensive flows, depicted in Figure 4. Going from top down, we depict separately the two main scenarios of data-intensive flows studied in this paper (i.e., ETL and ETO). Going from left to right, the first part of Figure 4 (i.e., A, E) depicts the characteristics of the *data extraction* stage in terms of the complexity that different input data types bring to a data-intensive flow; then the following part (i.e., B, F) covers the characteristics of the *data transformation* stage; the penultimate part (i.e., C, G) covers the *data delivery* stage, while the last (most right) part (i.e., D, H) covers *querying*. Being rather a holistic field (i.e., taking into account the complete data-intensive flow), the *optimization* spans all stages of data-intensive flows and it is depicted at the bottom of Figure 4.

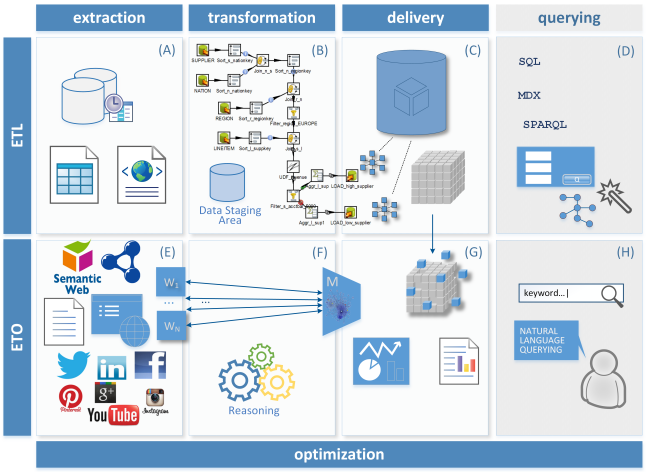

Fig. 4. Study setting for data-intensive flows

3.3 Phase II (Analyzing the characteristics of data-intensive flows).

This phase included the review of the works that scrutinize the characteristics of data-intensive flows in both previously defined scenarios, i.e., *ETL* and *ETO*. This phase aimed at characterizing data-intensive flows in terms of the features that best indicate the movement toward the *next generation BI* setting.

To this end, we performed an incremental analysis of the included works to discover the features of data-intensive flows they have tackled. We started from the papers that individually covered the *traditional* and the *next generation BI* settings. The identified features are then translated into the dimensions for studying data-intensive flows (see Figure 5). As new dimensions are discovered, the related papers are reconsidered to analyze their assumptions regarding the new dimensions. Each discovered dimension determines the levels, supported or envisioned by analyzed approaches, in which these approaches attain the corresponding feature of data-intensive flows. Eventually, we converged to a stable

set of dimensions, which can be further used for studying and classifying the approaches of data-intensive flows.

In Section 4, we discuss in more detail the process of discovering dimensions for studying data-intensive flows, and further provide their definitions.

3.4 Phase III (Classification of the reviewed literature).

In this phase, we further extend the study to the works that more specifically cover the previously discussed areas of data-intensive flows (i.e., *data extraction*, *data transformation*, *data delivery*, and *data flow optimization*). We classify the reviewed approaches using the previously defined dimensions, which build our study setting (see Figure 5), and present the results of this phase in Sections 5 - 8. We summarize the classified approaches of the three main stages (i.e., *data extraction*, *data transformation*, and *data delivery*) respectively in Tables 1 (page 18) - 3 (page 28), and the optimization approaches in Table 4 (page 31). We mark the level of the particular dimension (i.e., challenge) that each approach achieves or envisions (i.e., `Low`, `Medium`, or `High`)[6].

In addition, we also classify the approaches in Tables 1 - 4 based on the fact if they are potentially applicable in *ETL*, *ETO*, or both *ETL* and *ETO* scenarios.

Finally, for each reviewed approach we define the *technology readiness level*, focusing on the first four levels of the European Commission scale [23].

- `TRL1` (*"basic principles observed"*), refers to work that either based on practical use cases or reviewed literature observes the basic principles that should be followed in practice (e.g., guidelines, white or visionary papers)
- `TRL2` (*"technology concept formulated"*), refers to work that provide theoretical underpinnings of the studied area, which are not always directly applicable in practice, but represent an important foundation for principles that should be followed in practice (e.g., the database theory works on data exchange and data integration).
- `TRL3/TRL4` (*"experimental proof of concept"*/ *"technology validated in lab"*), refers to the system-oriented work that provide the proof of concept solution for an observed principle from the previous two levels, validated either over synthetic (`TRL3`) or real-world use cases (`TRL4`).

4 Defining dimensions for studying data-intensive flows

For each area in the outlined study setting for data-intensive flows (Figure 4), we discuss in more detail, and further provide the definitions of the dimensions through which the reviewed works on data-intensive flows are analyzed (see Figure 4). Then, in the following sections 5 - 8, we discuss in more detail the works specifically covering each of the studied areas.

[6] The exception to this are the approaches from the data flow optimization area, for which we introduced levels that more precisely describe the consequences of their placement inside the corresponding dimensions. Moreover, in the cases when the approach is completely independent of the level for a particular dimension, we mark it as non-applicable (`N/A`).

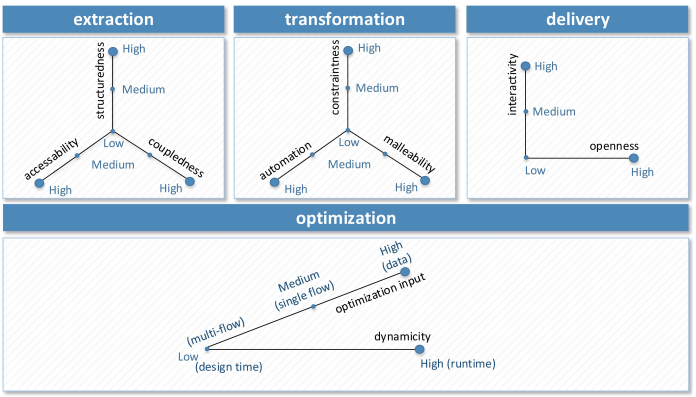

Fig. 5. Dimensions for studying data-intensive flows

4.1 Data Extraction

The most commonly discussed challenge when extracting data (e.g., [1,53]) is related to the format in which the data are provided, i.e., *structuredness*.

Structuredness determines the level, in which data in data sources under analysis follow a certain model, constraints or format. It spans from highly structured data that follow strictly defined models and ensures certain constraints over data (`High`), like relational (see top left of Figure 4); then semi-structured data that are represented in a repetitive [47], standard and easily parsable format, but that do not enforce strong constraints over data (`Medium`), like XML, CSV, RDF (see middle left in Figure 4); and unstructured data in a free-form (textual or non-textual) that require smarter techniques for extracting real value from it (`Low`), like free text, photos, x-rays, etc. (see bottom left of Figure 4). □

Other characteristics of this stage are related to the degree in which BI applications need to be coupled with source systems when extracting the data, i.e., *coupledness*, and the reliability of accessing these systems, i.e., *accessibility*.

Coupledness determines the level, in which data-intensive flows depend on a specific knowledge or components obtained from data sources under analysis. It spans from typical ETL processes that consolidate organization's ("in-house") operational sources with predictable access policies (e.g., DBMS, ERP systems; see top left of Figure 4), using extraction algorithms (e.g., incremental DB snapshots) that strongly depend on information from source systems (`High`), e.g., materialized views DW solutions, triggers, source components modifications; then the systems that use logs, timestamps or other metadata attached to data sources (`Medium`); and the scenarios where we cannot expect any in-advance knowledge about the data sources, but the system needs to discover and integrate them on-the-fly [2] (`Low`), e.g., Web data, Link Open Data. □

Accessability determines the level, in which one can guarantee a "non-stop" access to certain data sources. It spans from "in-house", highly available data (`High`), like ERP, ODS systems; then the external data usually provided by data

providers that under SLAs can guarantee certain accessibility to data (`Medium`); and external data sources that are completely out of the organization's control (`Low`), like open, situational, or Web data. □

4.2 Data Transformation

In this stage, the most common issue is related to the complexity of data transformations inside a flow (e.g., [96]), and more specifically to the degree in which we can automate the design of a data-intensive flow, i.e., *automation*. While the design of an ETL process is known to be a demanding task (e.g., [89]; see the example ETL process in the top middle of Figure 4) and its automation (even partial) is desirable and has been studied in the past (e.g., [68,79]), ETO depends on fully automated solutions for answering user queries at runtime (e.g., by means of reasoning; see Figure 4).

Automation determines the level, in which one can automate the design of data-intensive flows. It spans from the works that propose modeling approaches to standardize the design of data flows, especially in the context of ETL processes (`Low`), e.g., [99,92,3]; then approaches that provide guidelines and/or frequently used patterns to facilitate the design of a data-intensive flow (`Medium`), e.g., [53,98]; and approaches that attempt to fully automate the generation of data-intensive flows as well as their optimization (`High`), e.g., [79,24,19]. □

Other important characteristics that distinguish *ETO* from a traditional *ETL* process are the degree of data *constraintness* that a flow must ensure, and the flexibility (i.e., *malleability*) of a data-intensive flow in dealing with the changes in the business context.

Constraintness determines the level, in which data-intensive flows must guarantee certain restrictions, constraints, or certain level of data quality over the output data. It spans from fully constrained data, usually enforcing the MD model (MD integrity constraints [64]), and high level of data cleanness and completeness required to perform further data analysis, like OLAP (`High`); then, data-intensive flows that may provide ad-hoc structures for answering user queries (e.g., reports), without a need to enforce the full completeness and cleanness of data (`Medium`); and as an extreme case we consider the concept of *data lakes* where no specific schema is specified at load time, but rather flexible to support different analysis over stored and shared data at read-time (`Low`). □

Malleability determines the level in which a system is flexible in dealing with the changes in the business context (e.g., new/changed data sources under analysis, new/changed/removed information requirements). It spans from the traditional DW settings where data sources as well as information requirements are static, typically gathered in advance. and are added manually to the analysis only at design time, while any change would require the redesign of a complete data-intensive flow (`Low`) [53]; then systems that tackle the incremental evolution of data-intensive flows in front of new requirements and data sources (`Medium`) [49]; and dynamic approaches that consider discovering new, usually external data at runtime (`High`) [2]. □

4.3 Data Delivery

For delivering the transformed data at the target, we have identified two main characteristics that distinguish ETL from ETO, namely the *interactivity* of data delivery as perceived by the end-user; and the *openness* of the delivered data, which refers to the degree in which the approaches assume the delivered information to be complete (i.e., *closed* vs. *open world assumption*).

Interactivity determines the level in which a system interacts with the end-user when executing a data-intensive flow and delivering the data at the output. It spans from traditional ETL processes that typically deliver the data (i.e., materialize the complete data to load a DW) in a batched, asynchronous process, without having an interaction with an end-user (`Low`), then approaches that based on the overall cost, select data to be partially materialized (e.g., loaded in a batch to materialized views), and those that are queried on-the-fly (`Medium`); and finally completely interactive approaches that assume on-the-fly data flows which deliver the data to answer user queries for immediate use only, e.g., for visualization (`High`). □

Openness determines the level, in which the delivered data are considered *open* to different interpretations. It spans from *closed-world assumption* approaches typical for traditional databases, where the data are considered complete and any answer to a user query is determined (`Low`), to *open-world assumption* approaches, where due to the assumption that data may be incomplete, an answer to a user query can be either determined if there exist data that can prove such an answer, or "unknown" in the case where there is no data to determine its truthfulness (`High`). □

4.4 Optimization of data-intensive flows

Finally, we also discuss the low *data latency* as an important requirement for today's data-intensive flows.

Optimizing data flows has been one of the main topics in database research [48]. In this context, we discuss the following two aspects.

Optimization input. This dimension refers to the level at which the optimization is provided. It spans from optimizing the way that input `data` is stored and processed (e.g., data fragmentation) in order to achieve optimal execution (e.g., parallelizing data flow execution); then optimizing the execution of `single data flows` by means of modifying the execution of the data flow (e.g., operation reordering, different implementations) to achieve the optimal execution of a data flow; and finally the overall optimization of a `multi-flow`, where the goal is to achieve the optimal execution for a set of data flows, rather than optimizing the execution of a single flow. □

Dynamicity. Another dimension for studying the optimization of data-intensive flows relates to the overhead introduced by the optimization and thus determines the level of *dynamicity* of the data flow optimization process. In the traditional DW systems, the design and optimization of ETL processes is done at the `design time`, once while the process is then executed periodically,

many times. This obviously allows for a higher overhead of the optimization process and taking into account different metadata (e.g., statistics from the previous executions of the same flow). On the other hand, an *ETO* flow must be optimized at `runtime`, when the analytical query is issued, which introduces additional challenges into the optimization techniques, especially regarding the way the flow and data statistics are gathered and exploited for optimization. □

5 Data Extraction

In the initial stage of data-intensive flows, the source systems are identified and accessed for extracting the relevant data. In the data extraction stage, we focused on analyzing the three following challenges that characterize the shift towards the *next generation BI* settings, namely *coupledness*, *accessibility*, and *structuredness*, which are subsequently discussed in the following subsections.

5.1 Structuredness

The seminal work on DW systems (e.g., [46]), although mentioned external unstructured data as an important opportunity for building DW system, have not in particular tackled the challenges that they bring to the design of the ETL pipelines. Furthermore, some of the first (purely relational) approaches for the DW system design in the literature (i.e., using materialized views in [90]), as well as the later extraction techniques (e.g., [57,62]) assumed purely relational data sources, thus only supported `High` structuredness of input data. [53] considers `Medium` stucturedness of input data, by providing the practical guidelines for accessing external data in Web logs or flat files, and semi-structured data (i.e., XML; see Figure 4(A)).

Example. In our running example, the ETL flow depicted in Figure 4 reads data from the transactional systems supporting daily sales operations. Notice that besides the challenges that heterogeneity in data sources (both structural and semantic [87]) brings to the design of an ETL pipeline, well-structured data sources do not require any special technique for obtaining the data before the transformation and cleaning starts. However, today, using a diversity of external and often unstructured data sources is becoming inevitable and thus the techniques for extracting such data have attracted the attention of both academy and industry. In our example scenario, we can see that introducing unstructured data (e.g., free text `reviews` from the Web) into the analytical processes of the company (see Figures 2 and 3) required additional data processing for extracting relevant information from these sources. Specifically, `Sentiment Analysis` based on natural language processing (NLP) is performed over textual data from the Web to extract customer opinions about the `items` and the campaign. □

In research, different techniques (e.g., text mining [28], NLP [58,29], sentiment analysis) are proposed to discover data patterns and extraction rules for extracting relevant data from natural language documents and transform unstructured data into more explicitly structured formats [10] (e.g., graphs, trees,

relational model). There, approaches are hence able to deal with the `Low` structuredness of input data. However, at the same time, they may assume a considerable latency overhead to the execution of the complete data pipeline and thus introduce an additional challenge to data-intensive flows. Interestingly, the linked data movement [7], on the other side, proposes that large amounts of external data are already provided in more structured (semi-structured) formats and semantically interlinked (e.g., using RDF), in order to facilitate the situation-aware analysis [63] and data exploratory actions [2]. These approaches assume `Medium` structuredness of input data, ready for the analytical processes carried out by data-intensive flows.

5.2 Coupledness

First relational approaches for designing a DW by means of a set of materialized views (e.g., [90]) in general allowed very efficient refreshments processes, by applying the well-known view maintenance techniques, to either compute incremental changes in the sources or a complete "rematerialization" of a view. Both approaches issue queries (*maintenance queries*) over the data sources, extract the answer, and load the corresponding views. Such approaches, however, required a `High` coupledness to source systems and soon became unrealistic to support the demands of enterprises to include a variety of external data sources into their analytical processes.

Example. As we can see from our example scenario, the retail company initially relied mainly on the internal data sources (e.g., the information about the `sold items`), which are periodically transferred to a central DW (see Example 1.1). To lower the data transferred in every execution of the ETL flow, the designers have built the flow to only extract the `sales` and `item` data that are inserted to the sources after the last ETL execution (i.e., snapshot difference). For efficiently finding the differences in two snapshots of the source data, the tight (`High`) coupledness to the considered data sources is needed. On the other side, in the scenario illustrated in Examples 2.1 and 2.2 (i.e., Figures 2 and 3, respectively), some data sources (i.e., `item reviews` from the Web) are not under the control of the company and moreover they may not be known in advance as their choice depends on the current user needs. Thus, obviously we cannot depend on having strong knowledge of these kinds of data sources. □

In the context of modern ETL processes, in [96], the author revisits the approaches for finding the difference of the consecutive snapshots of source databases (e.g., [57,62]). However, these snapshot techniques (e.g., timestamps, triggers, interpreting the source's logs) still required certain control over the known source systems (i.e., `Medium` coupledness). Web-oriented systems have further imposed more relaxed environments for extracting data located on disparate Web locations. The most common solution introduced for performing data integration from disparate sources includes building specialized software components, called *wrappers*, for extracting data from different Web sources (e.g., [32]; see Figure 4(E)). Wrappers are typically used in combination with another component (namely *mediator*, see Figure 4(F)), which, based on a user query, invokes

individual wrappers, and combines (i.e., integrates) data they return to answer the input query. However, the wrapper/mediator architecture still requires a `Medium` coupledness, as wrapper design highly relies on the specific technology of data source systems, while the changes in the source systems typically require reconsidering the wrappers' design. Finally, as we mentioned above, to make the enormously growing data volumes on the Web available and potentially useful for the enterprise analytical and data discovery actions, the linked and open data movement (e.g., [7]) has proposed a completely uncoupled environment (i.e., `Low` coupledness) with the general idea of having huge amounts of distributed data on the Web semantically interlinked and preferably provided in easily parseable formats (e.g., XML, RDF), see Figure 4(E). Approaches that argue for such `Low` coupled scenarios, envision architectures that can take the advantage of existing logic-based solutions for enabling data exploration actions over the external sources [2], and provide more context-aware data analysis [1].

A separate branch of realated research that strongly argues for `High` decoupling of data sources in data-intensive flows is Complex Even Processing (CEP) [14]. The idea here is on enabling on-the-fly processing and combining of data coming in greater speed and typically from external data sources, with the goal of detecting different correlations or anomalies happening in the "external world". Thus, the CEP systems typically decouples from the technical level information of the data sources (e.g., sensor readings), and rather aims at detecting events at the application level (e.g., correlations, anomalies). CEP is rather extensive and separate fields of research, and to this end, we here give its high level overview in terms of our analysis dimensions, while for the specific approaches and applications we refer the reader to the survey in [14], which compares and studies in detail the state of the art approaches in this field.

5.3 Accessability

The practical guidelines for efficiently building an ETL process in [53] proposes a pre-step of profiling data sources for quality, completeness, fitness, and accessibility. Apart from transactional data sources, dominant in traditional DW scenarios [46], with typically `High` accessability or at least predictable access policies (e.g., nightly time windows), nowadays a vast amount of potentially useful data for an enterprise is coming from remote data sources, over the global networks, like forums, social networks, and Web in general, [1], see Figure 4(E).

Example. Going back to our example scenario, we can notice that in the traditional DW environment, the company builds the system based on the previously elicited business needs and accordingly incorporates internal data sources (e.g., `item` and `sales`) into their analytical processes. ETL flows (e.g., see Figure 1) are designed and tested in advance for periodically extracting the data from pre-selected data sources, relying on the `High` or predictable availability of these sources. Conversely, the ETO flows in Figures 2 and 3 cannot rely on accessing the data sources at all times, due to remote access (e.g., Web and social networks) and moreover as they can be selected on-the-fly. □

Even though in the linked (open) data movement information about quality of external data are envisioned to be in the form of catalogs [7], the access to these data at any moment still cannot be guaranteed (`Low` accessability), which brings a new challenge to the process of data extraction in this scenario. In this context, the authors in [1] study the concept of situational data, which are usually external to an organization control and hence without a guaranteed access, but which in fact play an important role in today's context-aware decision making. The authors thus propose a "*data as a service*" solution, where envisioned systems will have a registry of possible data providers, and using Web service interface partially automate the process of finding the most appropriate and currently accessible data source.

Table 1. Classification of data extraction approaches

Data extraction					
ETL vs. *ETO*	**Approaches** AUTHORS, YEAR, [***NAME***,] REFERENCE	TRL	**Dimensions**		
			struct.	*access.*	*coupl.*
ETL	Inmon, 1992, [46]	1	High	High	High
	Theodoratos & Sellis, 1999, [90]	2			
	Labio et al. 1996, [57] Lindsay et al. 1987, [62]	3	High	N/A	Medium
	Kimball & Caserta, 2004, [53]	1	Medium	Medium	High
ETO	Feldman & Sanger, 2007, [28] Buneman et al., 1997, [10]	1	Low	N/A	N/A
	Laender et al., 2002, [58]	1	Low	Low	Medium
	Bizer et al., 2009, ***Linked Data***, [7] Cugola and Margara, 2012, ***CEP***, [14]	1	Medium	Low	Low
	Abelló et al., 2013, ***Fusion Cubes***, [1] Abelló et al., 2015, [2]	1	Low	Low	Low
	Garcia-Molina et al., 1997, ***TSIMMIS***, [32]	4	Medium	High	Medium

5.4 Discussion

We summarize the results of studying the challenges of *data extraction* stage (i.e., the classification of the representative approaches) in Table 1. As expected, we have found more matured (i.e., $TRL \geqslant 2$) works dealing with this stage in the *traditional BI* setting (i.e., *ETL*), considering tighter *coupledness* to source systems, relying on high *accessibility*, and expecting *structured* data. Several approaches have opened the issue of dynamically accessing external and unstructured data, focusing mostly on data coming from the Web, while the majority considered structured (relational) or at most semi-structured (XML) data.

Data extraction is however an important stage to be reconsidered for today's data-intensive flows, especially taking into account new loosely coupled BI environments [1]. The main challenges of these (mostly envisioned) ecosystems with low coupledness relate to the fact that data sources are outside of the organization control, and often not even known in advance. Thus, the efficient techniques to discover the relevant data must be deployed. We can benefit here from the

known techniques proposed to explore the contents on the Web (e.g., *Web crawling*). Moreover, being external data sources, the systems become very sensitive to very probable variability of data formats, as well as the undefined semantics of data coming from these sources. To overcome the semantics heterogeneity gap between the data sources, and to automate discovering and extracting the data from them, we propose to use the semantic-aware exploratory mechanisms [2].

Furthermore, as we can see from our example scenario, specialized data processing (e.g., *natural language processing*, *sentiment analysis*, *text mining*) should be also considered to extract the relevant information from these, often unstructured, data sources. However, such complex data transformations typically affect the data latency in a data-intensive flow, hence in most of the current approaches this kind of input data transformation has been considered as part of a pre-processing step. An alternative to this, following the principles of *linked (open) data* [7], is to have data published in at least semi-structured formats (e.g., XML, CSV, RDF), which largely facilitates their further exploitation.

6 Data Transformation

After data are extracted from selected (often heterogeneous) sources, the flow continues with transforming the data for satisfying business requirements and considered quality standards. Data transformation is characterized as the main stage of a data-intensive flow by most of the approaches [53,96]. The main dimensions we analyze in the data transformation stage are *automation*, *malleability*, and *constraintness*.

As previously mentioned, from early years, managing heterogeneous data has brought the attention of database community, and some fundamental works stem from the database theory field (i.e., *data integration* (e.g., [59,94]) and *data exchange* (e.g., [27])) to tackle this problem from different perspectives.

Both data exchange and data integration problems are based on the concept of *schema mappings*, which in general can be seen as assertions that define the correspondences between source and target schema elements.

In general, a parallelism can be drawn between the theoretical problems of data exchange and data integration, and what we today consider as data-intensive flows. Similar observation has been discussed in [96]. The author compares data exchange to the traditional DW setting, where data transformations in the ETL pipeline can be generally seen as schema mappings of the data exchange setting. However, as also noted in [96], schema mappings, as defined by these theoretical approaches, are typically limited to simple transformations over data and do not efficiently support typical transformations of data-intensive flows (e.g., grouping, aggregation, or black-box operations), nor the diversity of data sources (i.e., only relational or XML data formats have been considered).

6.1 Malleability

Data-intensive flows, as other software artifacts, do not lend themselves nicely to evolution events, and in general, maintaining them manually is hard. The

situation is even more critical in the *next generation BI* settings, where on-the-fly decision making requires faster and more efficient adapting to changed domain context, i.e., changed data sources or changed information needs.

For considering the former problem, we revisit the foundational works on data exchange and data integration, which introduced two main approaches for schema mappings, i.e., *global-as-view* (GAV) and *local-as-view* (LAV) [30,59].

In the GAV approach, the elements of the global schema are characterized in terms of a query over the source schemata, which further enables less complex query answering by simply unfolding global queries in terms of the mapped data sources. However, GAV mappings lack flexibility in supporting the evolution of data source schemata, as any change on the sources may potentially invalidate all the mapping assertions (i.e., `Low` malleability). An example of this approach is the wrapper/mediator system [32].

Example. As we discussed, in Scenario 1, the company elicits the business needs prior to designing the DW and ETL flows (e.g., see Figure 1). In the case a new data source is added, the redesign of the system is performed offline before the ETL flows are run again. However, notice that in the second scenario (see Examples 2.1 and 2.2) the flexibility of the system for adding new data sources must be supported in an "instant" manner, as business needs are provided on-the-fly and often require a prompt response. □

As opposed to GAV, LAV schema mappings characterize the elements of source schemata in terms of a query over the global schema. LAV mappings are intuitively used in the approaches where changes in dynamic data source schema are more common (e.g., [54]) as it provides `High` malleability of the data integration systems. We can thus observe that the LAV approach fits better the needs of the *next generation BI* setting, where the variability and number of data sources cannot be anticipated (e.g., [1,2]). However, the higher flexibility of LAV mappings brings the issues of both, the complexity of answering the user queries and the completeness of the schema mappings. Generally, in LAV, answering user queries posed in terms of a global schema implies the same logic as answering queries using materialized views, which is largely discussed as a computationally complex task [42].

Several approaches further worked on generalizing the concept of schema mappings by supporting the expressive power of both LAV and GAV, i.e., *both-as-view* (BAV) [65], and *global-and-local-as-view* (GLAV) [30].

However, as we discussed before such approaches are hardly applicable to the complex data-intensive flows. In the context of the traditional DW systems, some works have studied the management of data-intensive flows (i.e., ETL process) in front of the changes of data source schemata. In [70] the authors propose a framework for impact prediction of schema changes for ETL workflow evolution. Upon the occurred change, the ETL flow is annotated with (pre-defined) actions that should be taken, and the user is notified in the case that the specific actions require user involvement. Other approaches (e.g., [49]) have dealt with automatically adapting ETL flows to the changes of user's information needs. For each new information requirement, the system searches for the way to adapt

the existing design to additionally answer the new requirement, by finding the maximal overlapping in both data and transformation. Lastly, some approaches have also dealt with the problem of adapting DW systems to the changes of the target DW schema. Being a "non-volatile collection of data" [46], the evolution changes of the target DW schema are typically represented in terms of different versions of a DW (i.e., multiversion DW). In particular, the most important issue was providing a transparent querying mechanisms over different versions of DW schemata (i.e., cross-version querying; [67,37]). For instance, a solution proposed in [37] suggested keeping track of change actions to further enable answering the queries spanning the validity of different DW versions. These approaches provide a certain (`Medium`) level of malleability for data-intensive flows, but still lack the full automation of the evolution changes or applicability in the case of unpredictable complexity of data transformations.

In addition, after data warehousing was established as a de facto way to analyze historical data, the need for more timely data analysis has also emerged in order to support prompter detection of different anomalies coming from data. This led researches to rethink the current DW architecture and make it more malleable to combine both traditionally mid-term and long-term, with "just-in-time" analysis. This brought the idea of (near) real-time data warehousing systems. Several approaches discussed the main requirements of such systems and proposed architectural changes to traditional DW systems for satisfying these new requirements (e.g., [8,97]). For instance, besides the main requirement of data freshness, [97] has also indicated minimal overhead of the source system and scalability in terms of input data sources, user queries, and data volumes, as relevant for these systems. They however pointed out the contradiction between users need for maximal data freshness and completeness, and the high overhead of the traditional DW workflows that often require costly data cleaning and transformations. To this end, the approaches in [8] and [97] discuss both conceptual and technological changes that would balance the delays in traditional ETL processes. In practice, SAP Business Warehouse [66] is an example of such system. It provides certain flexibility to traditional DW systems for enabling on-the-fly analysis at different levels of data, i.e., summarized and loaded to a DW, consolidated operational data (operational data store), or even directly over the transactional data. Their goal is to enable more real-time data warehousing and a possibility of also including fresh, up-to-date trasnactional data to the analysis. Even though the (near) real-time DW approaches bring more malleability (`Medium`) to data analysis by combining historical and on-the-fly analysis, included data are still typically coming from the in-house and predefined data sources.

6.2 Constraintness

What further distinguishes data exchange [27,55] from the original data integration setting, is that the target schema additionally entails a set of constraints that must be satisfied (together with schema mappings) when creating a target instance (i.e., `High` constraintness).

Example. In Figure 1, we can notice that for loading data into a DW, data-intensive flow must ensure a certain level of data quality to satisfy constraints entailed by the DW (e.g., `Remove Duplicates` in Figure 1 removes the repetitive `itemID`s for loading the `successFact` table into a DW). On the other side, data-intensive flows in the *next generation BI* settings (see Figures 2 and 3), due to their time constraints typically cannot afford to ensure full data quality standards, but is often sufficient to deliver partially cleaned (i.e., "right") data, at the right time to an end-user [22]. □

The work on generalizing schema mappings (GLAV) in [30] also discusses the importance of adding support for defining the constraints on global schema, but no concrete solution has been provided. In the data integration setting, although some works did study query answering in the presence of integrity constraints on global schema [12], (i.e., `Medium` constraintness), most of the prominent data integration systems (e.g., [32,54]) typically do not assume any constraints in the target schema (i.e., `Low` constraintness). Furthermore, as we discussed in the DW context, the design of an ETL process is affected by the integrity constraints typical in a dimensionally modeled DW schema (see Figure 4(C)).

When working with data from unstructured data sources, one may face two different problems: (1) how to extract useful information from data in unstructured formats and create more structured representation; and (2) how to deal with incomplete and erroneous data occurred due to lack of strict constraints in source data models. The latter problem becomes even more challenging when the target data stores entail strict constraints as we discussed above. While the former problem is usually handled by means of data extraction techniques discussed in the Section 5, the latter is solved at the data transformation stage, where data are cleaned to fulfill different quality standards and target constraints. As expected, such a problem has brought the attention of researchers in the data exchange (e.g., [25]) and data integration (e.g., [21]) fields. In the modern DW systems, target data quality and `High` constraintness is usually guaranteed as the result of the process called data cleaning. Data cleaning deals with different data quality problems detected in sources, e.g., lack of integrity constraints at sources, naming and structural conflicts, duplicates [75].

6.3 Automation

It is not hard to see from the previously discussed problem of data cleaning and the flows in the example scenario, that today's data-intensive flows require more sophisticated data transformations than the ones (mainly based on logics) assumed by fundamental approaches of *data exchange* and *data integration*. At the same time, higher automation of the data flow design is also required to provide interactive, on-the-fly, analysis.

Example. Loading a DW may require complex data cleaning operations to ensure the entailed constraints. Obviously, complete automation of the design of such data-intensive flows is not realistic and thus the designers in Scenario 1 usually rely on a set of frequent data transformations when building ETL flows (e.g., `Join`, `UDF`, and `Remove Duplicates` in Figure 1). But, in Scenario 2, such

an assisted design process is not sufficient, as the flows for answering users' on-the-fly queries (e.g., see Examples 2.1 and 2.2) must be created instantaneously. This, together with the requirement for lower data latency, restricted such flows to more lightweight operations (e.g., `Filter` or `Match` in Figure 2). □

Different design tools are available in the market and provide often overlapping functionalities for the design and execution of data-intensive flows (mostly ETL; see for example Figure 4(B)). The complexity and variability of data transformations has introduced an additional challenge to the efforts for providing a commonly accepted modeling notation for these data flows. Several works have proposed different ETL modeling approaches, either ad-hoc [99], or based on well-known modeling languages, e.g., UML in [92] or BPMN in [3,101]. However, these modeling approaches do not provide any automatable means for the design of an ETL process (i.e., `Low` automation). Some approaches (e.g., from UML [68], or from BPMN [4]) are further extended to support certain (i.e., `Medium`) automation of generating an executable code from the conceptual flow design, by means of model transformations (i.e., Model-driven design).

The design of an ETL process is on the other side described as the most demanding part of a DW project. As reported in [89] ETL design can take up to 80% of time of the entire DW project. In [53] the authors give some practical guidelines for a successful design and deployment of an ETL process, but without any automatable means (i.e., `Low` automation), still, a considerable manual effort is expected from a DW designer. In [98], the framework that uses the ad-hoc modeling notation from [99] is proposed to assist the design of ETL processes, along with the palette of frequently used ETL patterns (i.e., `Medium`).

Several approaches went further with automating the conceptual design of ETL processes. On the one hand, in [88], the authors introduced the design approach based on Semantic Web technologies to represent the DW domain (i.e., source and target data stores), showing that this would further enable automation of the design process by benefiting from the automatic reasoning capabilities of an ontology. [79], on the other hand, assumes that only data sources are captured by means of a domain ontology with associated source mappings. Both DW and ETL conceptual designs are then generated to satisfy information requirements posed in the domain vocabulary (i.e., ontology). Finally, [5] entirely rely on an ontology, both for describing source and target data stores, and corresponding mappings among them. Integration processes (ETL) are then also derived at the ontological level based on the type of mappings between source and target concepts (e.g., equality, containment). However, even though these approaches enable `High` automation of the data flow design, they work on a limited set of frequent ETL operations.

In parallel, in the field of data exchange, [24] proposes a tool (a.k.a. *Clio*) that automatically generates correspondences (i.e., schema mappings) among schemas without making any initial assumptions about the relationships between them, nor how these schemas were created. Such a generic approach thus supports `High` automation in creating different schema mappings for both data integration and data exchange settings. [19] went further to provide the inter-

operability between tools for creating declarative schema mappings (e.g., *Clio*) and procedural data-intensive tools (e.g., ETL). Still, such schema mappings either cannot tackle grouping and aggregation or overlook complex transformations typical in today's ETL processes. The *next generation BI* settings, however, cannot always rely on the manual or partially automated data flow design. Moreover, unlike ETL, ETO cannot completely anticipate end user needs in advance and thus besides the `High` level of automation, the design process must also be agile to efficiently react in front of new or changed business needs (e.g., [78]).

Table 2. Classification of data transformation approaches

Data transformation					
ETL vs. *ETO*	**Approaches** AUTHORS, YEAR, [***NAME***,] REFERENCE	TRL	**Dimensions**		
			autom.	*malleab.*	*constr.*
ETL	Fagin et al., 2003, ***Data Exchange***, [27] Kolaitis, 2005, [55]	2	N/A	N/A	High
	Rahm & Hai Do, 2000, [75]	1			
	Kimball & Caserta, 2004, [53]	1	Low	Low	High
	Vassiliadis et al., 2002, [99] Trujillo & Luján-Mora, 2003, ***UML-ETL*** [92] Wilkinson et al., 2010, ***xLM***, [101] El Akkaoui et al., 2012, ***BPMN-ETL*** [3]	2	Low	Low	High
	Muñoz et al., 2009, [68] El Akkaoui et al., 2013, [4] Vassiliadis et al., 2003, ***ARKTOS II***, [98] Papastefanatos et al., 2009, [70] Morzy & Wrembel, 2004, [67] Golfarelli et al., 2006, [37]	3	Medium	Medium	High
	Skoutas & Simitsis, 2007, [88] Bellatreche et al., 2013, [5]	3	High	Low	High
	Fagin et al., 2009, ***Clio***, [24]	4			
ETL & ETO	McDonald et al., 2002, ***SAP BW***, [66]	4	Medium	Medium	High
	Romero et al., 2011, ***GEM***, [79] Dessloch et al., 2008, ***Orchid***, [19] Jovanovic et al., 2012, ***CoAl***, [49]	3	High	Medium	High
ETO	Garcia-Molina et al., 1997, ***TSIMMIS***, [32]	4	High	Medium	Low
	Kirk et al., 1995, ***Information Manifold***, [54]	3	High	High	Low
	Romero & Abelló, 2014, [78] Abelló et al., 2014, [2]	1			
	McBrien & Poulovassilis, 2003, ***BAV***, [65] Friedman et al., 1999, ***GLAV***, [30] Calì et al., 2004, [11]	2	N/A	High	Medium

6.4 Discussion

As we have seen, in the *next generation BI* setting (i.e., *ETO*), where data sources are often external to the organization control and moreover discovered dynamically based on current user needs; a more flexible environment is needed for efficiently supporting adding new data sources to analytical processes. The local-as-view (LAV) schema mapping approaches are more suitable in order to support the required level of *malleability* [65,30]. In the LAV approach, plugging new data sources requires defining a mapping of the new source schemata to the global schema, without affecting the existing mappings. However, as we can see

in Table 2, currently, most of these techniques are still purely theoretical (i.e., $TRL = 2$), while the high complexity and intractability of LAV approaches have been widely discussed, and hence represent a serious drawback for using LAV mappings in near real-time BI environments.

On the other side, some approaches (e.g., [24]) have worked on automating the creation of such schema mappings, which can be widely applicable for supporting answering information requirements on-the-fly. Even though we notice the lower requirement for the cleanness and constraintness of output data in the *next generation BI* setting (see Table 2), which would typically result with lower complexity of a data flow, today's BI applications do require rather complex data analytics, which are typically not supported in the schema mapping approaches. Some approaches try to extend this by automating the flow design (e.g., [79,5]), but still with very limited and predefined operation sets (i.e., *ETL* & *ETO*). Therefore, automating the creation of more complex data-intensive flows (e.g., machine learning algorithms), by means of exploiting different input data characteristics or using metadata mechanisms is still lacking.

7 Data Delivery

After data from multiple sources are cleaned, conformed, and combined together, a data-intensive flow delivers the data in the format suitable to the user needs either for visualization, or further analysis and querying. In the data delivery stage, we focus on analyzing the two following dimensions, namely, *interactivity* and *openness*, subsequently discussed in the following sections.

7.1 Interactivity

One of the important decisions that should be made while building data-intensive flows is the interactivity of the system when delivering data at the output.

Example. Notice that the ETL flow in Figure 1 is designed to periodically transfer the complete data about the `item sales` in a batched back-end process, so that users may later analyze the subset of these data depending on their needs (e.g., slicing it only to the `sales` in the third quarter of the last). ETO flows in Figures 2 and 3, however, instantly deliver the data from the sources that are currently asked by the user (e.g., trends of the past weekend, and trending product `items` from the first week of March, respectively). Moreover, such ETO flows are typical examples of answering ad-hoc and one-time analytical queries, thus storing their results is usually not considered as beneficial. □

Going back again to the fundamental work on *data exchange*, the data delivery in this setting is based on computing a solution, i.e., a complete instance of the target schema that satisfies both, schema mappings and constraints of the target schema. The queries are then evaluated over this solution to deliver the answer to the user. However, due to incompleteness of data and/or schema mappings, there may be more than one, and theoretically an infinite number of valid solutions to the data exchange problem [27]. Answering user queries in such

a case would result in evaluating a query over all possible solutions (i.e., finding the *certain answer*). To overcome the obvious intractability of query answering in data exchange, a special class of solutions (*universal solutions*), having a homomorphism into any other possible solution, is proposed.

Like in the data exchange setting, materializing the complete data at the target for the purpose of later answering user queries without accessing the original data sources, is also considered in the later approaches for designing a data warehouse (see Figure 4(G)), i.e., `Low` *interactivity*. As we discussed in Section 5, a DW has been initially viewed as a set of materialized views (e.g., [90]). Similarly, in this case the database specific techniques (e.g., incremental view maintenance) are studied to minimize the size of the data materialized in each run of refreshment flows (maintenance queries). However, as DW environments have become more demanding both considering the heterogeneity and volume of data, it became unrealistic to consider a DW solely as a set of materialized views. In addition, many approaches have further studied the modeling and the design of a target DW schema, which should be able to support analytical needs of end users. This has resulted in the field of multidimensional (MD) modeling that is based on fact/dimension dichotomy. These works belong to a broader field of MD modeling and design that is orthogonal to the scope of this paper, and thus we refer readers to the survey of MD modeling in [77] and the overview of the current design approaches covered by Chapter 6 in [38].

Conversely, the data integration setting, as discussed in Section 6, does not assume materializing a complete instance of data at the target, but rather interactively answering individual user queries (e.g., through a *data cube* or a *report*; see Figure 4(G)) posed in terms of a global (virtual) schema (i.e., `High` interactivity). A query is reformulated at runtime into queries over source schemata, using schema mappings (e.g., [32,54,59,42,94]). Another examples of `High` interactivity are Complex Event Processing and Data Stream Processing systems. Besides the differences these systems have (see their comparison in [14]), the common property of these systems is that they provide on-the-fly delivery of data, with typically low latency, and for the one-time use only (e.g., monitoring stocks, fraud detection), without a need to materialize such data.

In [34], in the context of peer data management, a hybrid solution is proposed based on both data exchange and data integration. The `Medium` *interactivity*, by partially materializing data and using a virtual view over the sources (peers), has been proposed. To this end, the solution presents schema dependencies that can be used both for computing the core and query answering.

The "right" (`Medium`) level of *interactivity* is also discussed to be crucial in the *next generation BI* setting (e.g., [15]) where a partial materialization is envisioned for a subset of data with low latency and low freshness requirements (i.e., for which we can rely on the last batched ETL run). Following the similar idea, [74] proposes a framework for combining data-intensive flows with user query pipelines and hence choosing the optimal materialziation point (i.e., `Medium` interactivity) in the data flow, based on different cost metrics (e.g., source update rates and view maintenance costs). Another field of research also

follows the `Medium` level of *interactivity*, and proposes an alternative to traditional ETL processes, where row data are first loaded to the target storage, and later, typically on-demand, transformed and delivered to end-users, i.e., extract-load-transform (ELT). For instance, an example ELT approach in [100] proposes an ELT architecture that deploys traditional database mechanisms (i.e., hierarchical materialized views) for enabling on-demand data processing of fresher row data previously bulk loaded into a DW.

7.2 Openness

As we have seen, incompleteness in source data (especially in the today's Web oriented environments, see Scenario 2 in Section 3), brings several challenges to data-intensive flows. In addition, when integrating and delivering the data at the target, due to possibly incomplete (or non-finite) data, the choice between two main assumptions should be made, i.e., closed world assumption (CWA) or open world assumption (OWA). This choice depends on different characteristics of both, the considered data sources and the expected target. For the systems, like in-house databases or traditional data warehouse systems, where the completeness of data can be assumed, CWA is preferable since in general we do not anticipate discovering additional data (i.e., `Low` openness). On the other side, when we assume incomplete data at the sources of analysis, we can either follow CWA and create a single finite answer from incomplete data (e.g., by means of data cleaning process), or OWA which would in general allow multiple and possibly an infinite number of interpretations of the answer at the target, by considering also dynamically added data to the analysis [2] (i.e., `High` openness).

Example. In our example scenarios, in the *traditional BI* setting (Scenario 1), the analysis of the revenue share depends solely on the data about the `item sales`, currently transferred to DW by means of the ETL process depicted in Figure 1. On the other hand, the *next generation BI* setting in Scenario 2, should assume a more open environment, where at each moment depending on the end user needs (e.g., following trends and opinions about `items` as in Examples 2.1 and 2.2) the system must be able to dynamically discover the sources from which such an information can be extracted (e.g., `reviews` in forums). □

Similarly, [71] revisits two main paradigms for the Semantic Web: (1) *Datalog*, that follows the CWA, and (2) *Classical* (standard logics) paradigm that follows OWA. An important conclusion of this work is that the *Datalog* paradigm as well as CWA is more suitable for highly structured environments in which we can ensure completeness of the data, while the *Classical* paradigm and OWA provide more advantages in loosely coupled environments, where the analysis should not only be limited to the existing (i.e., "in-house") data.

Moreover, coinciding arguments are found in the fields of *data integration* and *data exchange*. Intuitively, CWA (i.e., `Low` openness) is more suitable assumption in data exchange, where query answering must rely solely on data transferred from source to target using defined schema mappings and not on the data that can be added later [61]. Conversely, more open scenario is typically expected in data integration systems [20], where additional data can be explored on-the-fly

and added to the analysis [2] (i.e., `High` openness). However, the high complexity of query answering under the OWA [71], raises an additional challenge to the latency of data-intensive flows, which is critical in *next generation BI* systems.

Table 3. Classification of data delivery appraoches

Data delivery				
ETL vs. *ETO*	**Approaches** AUTHORS, YEAR, [***NAME***,] REFERENCE	TRL	**Dimensions** *interac.*	*open.*
ETL	Golfarelli & Rizzi, 2009, [38] Fagin et al., 2005, ***Data Exchange***, [27] Libkin, 2006, [61] Theodoratos & Sellis, 1999, [90]	2	Low	Low
ETL & ETO	Dayal et al., 2009, [16]	1	Medium	Low
	Qu & Dessloch, 2014, [74] Waas et al., 2013, ***ELT***, [100]	3		
	Giacomo et al., 2007, [34]	2	Medium	Medium
ETO	Cugola & Margara, ***CEP***, 2012, [14]	1	High	Medium
	Abelló et al., 2013, ***Fusion Cubes***, [1] Abelló et al., 2015, [2]	1	High	High
	Lenzerini, 2002, ***Data Integration***, [59] Halevy, 2001, [42] Ullman, 1997, [94] Doan et al., 2012, ***Data Integration*** [20]	2		
	Garcia-Molina et al., 1997, ***TSIMMIS***, [32] Kirk et al., 1995, ***Information Manifold***, [54]	3		

7.3 Discussion

The outcome of studying the data delivery stage of data-intensive flows can be seen in Table 3. We observed that the same principles for data delivery (i.e., levels of the studied dimensions in Table 3) are followed in approaches of traditional data exchange [27] and DW settings [38] (i.e., *ETL*). At the same time, we also noticed that the similar principles of the data integration setting [59,20] are envisioned for *next generation BI* systems in some of the studied approachs [1,2] (i.e., *ETO*), while others propose a mixed approach [16] (i.e., *ETL & ETO*).

Such observations strongly indicated that the underpinnings for building a system for managing data-intensive flows in the *next generation BI* setting should be searched in the theoretical field of data integration.

Such a trend has been indeed followed in some of the recent approaches. For example, the idea of creating a unified view over relevant data sources (i.e., the main principle of the data integration setting), is revisited by some approaches by creating a common domain vocabulary and integrating it with existing data sources (e.g., [88,79]). There, the use of a domain ontology to reconcile the languages of business and IT worlds when building a BI system has been proposed. In [79], an ontology is used in combination with schema mappings to automatically generate ETL pipelines to satisfy information requirements previously expressed in terms of an ontology by an end user. The approach works with a predefined set of data sources which is, as suggested, suitable for building a DW

system, but as data in today's BI settings are coming from disparate and external data sources, the challenge of capturing their semantics under a common vocabulary brings additional challenges. To this end, Semantic Web technologies are discussed (e.g., [2]) as a solution both for capturing the semantics and further interactive exploration of data, facilitated by the automatic reasoning mechanisms.

8 Optimization of data-intensive flows

Optimizing the execution of data-intensive flows, is a necessity, especially taking into account the *next generation BI* setting that often requires the "right-time" delivery of information.

8.1 Optimization input

The problem of data flow optimization has been considered from early years of databases from different perspectives, where each of these perspectives may affect different parts of a data flow.

- `Data.` The optimization of a data flow execution can be achieved by transforming the structure of the original dataset (e.g., by means of data partitioning [52]). However, notice that simply transforming a dataset would not achieve a better performance, unless the execution model of a data flow is able to exploit such a transformation (e.g., distributed or parallel data processing [18]).
- `Data flow.` The most typical case considers the optimization of data flow execution by changing the way data are processed, while ensuring the equivalent semantics of the resulting dataset. Such techniques stem from the early years of databases, where minimizing data delivery time by changing the order and selecting the most optimal algorithm for operations applied over input data has been studied under the name of *query optimization* [48]. Moving to the DW environment, which assumes more complex data transformations than the ones in relational algebra, has opened a new field of study dealing with optimizing *ETL* processes (e.g., [83]). In fact, similar principles to those introduced in query optimization (i.e., generating semantically equivalent execution plans for a query by reordering operations, and then finding a plan with a minimal cost) have been applied in [83] and extended to the context of *ETL* flows.
  Another work [44,76] has based operation reordering (i.e., plan rewrites) on automatically discovering a set of extensible operation properties rather than relying solely on algebraic specifications, in order to enable reordering of complex ("black-box") operators. While low data latency is desirable for *ETL* processes, due to limited time windows dedicated to the DW refreshment processes, in the *next generation BI* setting, having data-intensive flows with close to zero latency is a must. Other techniques include: choosing the

optimal implementation for the flow operations [93], selecting the optimal execution engine for executing a data flow [85,56], data flow fragmentation and pipelining [52,86].

– `Multi-flow.` In other scenarios, especially in the case of shared execution resources, the optimization goal may suggest optimizing the overall execution of a set of data flows, rather than only the execution of an individual flow. Approaches that deal with this problem fall in two categories. On the one hand, some approaches assume having a detailed knowledge of included data flows and thus try to exploit it and optimize the overall execution, by means of finding shared parts of data workloads and reusing common execution and data [80,35,49]. Other approaches however assume only a high level knowledge of included data flows (e.g., input data size, execution time, high-level flow complexity, time constraints for the flow execution; [73]). In such cases, the optimization of data flows proceeds by selecting the best scheduling for the execution of data-intensive flows, while the further optimization of individual data flows is left to an engine-specific optimizer [86].

8.2 Dynamicity

While the challenges due to the higher complexity of data transformation has been largely addressed [83,44], proposed cost-based techniques often require certain statistics metadata available for a given data flow in order to find the optimal configuration. However, this is typically not the case and gathering and managing such statistics is not an easy task [41]. [41] proposes a statistics collection framework, by defining a set of necessary statistics, as well as gathering methods. However, this approach although powerful assumes a case of ETL process flow, where data flows are typically designed and optimized in advance (i.e., at `design time`), while the statistics gathering depends on the previous execution of the same ETL process.

Notice that the majority of the optimization approaches discussed in the previous subsection also assume a static case (see Table 4), where data flows are optimized once, at `design time`, and then executed many times. The exception to this are approaches that besides statically optimizing a data flow, also provide dynamic optimization of data flow executions in terms of `runtime` scheduling (i.e., [86,73,52]).

Some optimization approaches however propose on-the-fly gathering of statistics, more suitable for the next generation data flow setting, and applying data flow optimization steps at `runtime`. The approach in [17] proposes performing micro-benchmarks for building models to estimate the costs of operations using different implementations or executing them on different engines. They show how to deal both with the conventional (relational algebra) operators as well as with complex data transformations typical for the next generation data flows (e.g., sentiment or text analysis). The importance of using more accurate statistics for optimizing data flows in dynamic, cloud-scale environments has been also discussed in [9]. To deal with uncertainty when optimizing running data flows they propose an approach that continuously monitors the execution of data flows

at `runtime`, gathers statistics, and re-optimizes data flows on-the-fly to achieve better performance. The focus here is however on the distributed computational model, where execution times are often higher than in the centralized systems due to necessary synchronization costs, thus the re-optimization overheads are typically considered as negligible.

Table 4. Classification of data flow optimization approaches

Data flow optimization				
ETL vs. *ETO*	**Approaches** AUTHORS, YEAR, [***NAME***,] REFERENCE	TRL	**Dimensions** *input*	*dynamicity*
ETL	Simitsis et al., 2005, [83] Hueske et al., 2012, [44] Rheinlnder et al., 2015, ***SOFA***, [76] Tziovara et al., 2007, [93] Simitsis et al., 2005, [85] Kougka et al., 2015, [56] Halasipuram et al., 2014, [41]	3	Data flow	Design time
	Giannikis et al., 2014, ***SharedDB***, [35] Jovanovic et al., 2016, ***CoAl***, [49]	3	Multi-flow	Design time
ETO	Karagiannis et al., 2013, [52]	3	Data & Data flow	Runtime
	Dayal et al., 2011, [17] Bruno et al., 2013, [9]	3	Data flow	Runtime
	Jarke & Koch, 1984, [48]	2		
	Simitsis et al., 2013, ***HFMS***, [86]	2	Multi-flow	Runtime
	Polo et al., 2014, [73]	3		

8.3 Discussion

In Table 4, we summarize the outcome of studying data flow optimization approaches in this section. It is easy to see that a static (design time) optimization of data flows has been largely studied in academia. While most approaches worked on the problem of extending traditional query optimization techniques [48] to support more complex data flow operations [83,44], they typically overlook the importance of having the needed statistics of input data and data flow operations to perform cost-based data flow optimization. Such design time optimization approaches require higher overhear and are hence mostly applicable to the traditional BI settings (i.e., *ETL*). Some of the recent approaches insist on the importance of having accurate statistics for creating an optimal execution of a data flow, both for design time [41] and runtime scenarios [9]. Still, the challenges for efficiently gathering and exploiting such statistics metadata for optimizing data-intensive flows remain due to the required close to zero overhead of an optimization process and the "right-time" data delivery demands in the *next generation BI* settings (i.e., *ETO*). To this end, the existing algorithms proposed for efficiently capturing the approximate summaries out of massive data streams [60], should be reconsidered here and adopted for gathering approximate statistics for data-intensive flows over large input datasets.

9 Overall Discussion

Finally, in this section, we summarize the main observations made from the results of our study and propose the high level architecture for managing the lifecycle of data-intensive flows in the *next generation BI* setting. We also give further directions for the topics that require special attention of the research community when studying data-intensive flows.

We have observed some general trends in studying the fields related to data-intensive flows. We focused on the fields of data exchange, data integration, as well as ETL, and ETO. The need for managing heterogeneous data has appeared ever since the database systems start being more broadly used (e.g., federated databases in the 80's [43]). Besides, even though the system in [82] from the 70's is argued to be the first approach that followed the principles of data exchange, the data exchange problem has not been formally defined until the early 00's [26]. Likewise, the problem of data integration is studied from the 90's [32,94], while the strong theoretical overview of the field is given in the early 00's [59]. Along with these theoretical works, the concept of the traditional DW setting was defined by Bill Inmon in the early 90's [46]. ETL, as a separate and rather complex process, however, appeared in the late 90's and the early 00's to replace simple refreshment processes for a DW modeled as a set of materialized views [96]. We can, however, notice the disparity among the trends of studying these fields in the past, showing that they have focused on solving isolated issues.

In the recent years, business environments became more complex, dynamic and interconnected, hence more interactive analytic systems to support daily decision making upon the combination of various (external or internal) data sources, have become a necessity. Moreover, as discussed throughout this paper, today's BI environments require efficiently combining these individual solutions for the problem at hand. To this end, in this paper, we have given a unified view of data-intensive flows, focusing on the challenges that *next generation BI* setting has brought. Currently, even though many works under different names (i.e., from different perspectives) have envisioned and/or proposed conceptual frameworks for *next generation BI* ecosystems (e.g., [1,6,15,17,22,63]), we still lack an end-to-end solution for managing the complete lifecycle of data-intensive flows. Going back to Tables 1, 2, and 3, we can observe a certain overlapping of levels of different dimensions between the theoretical problem of *data exchange* and data warehousing approaches (i.e., *ETL*), as well as between *data integration* and data-intensive flows in the *next generation BI* setting (i.e., *ETO*).

9.1 Architecture for managing the lifecycle of data-intensive flows in next generation BI systems

We additionally observed in Tables 1 - 3 that the majority of works supporting the idea of the *next generation BI* setting in fact belong to level 1 of technical readiness ($TRL = 1$), as they are mostly visionary works that analyze the challenges of the *next generation BI* setting from different perspectives. However, we still lack a complete picture of all the aspects of data-intensive flows in this

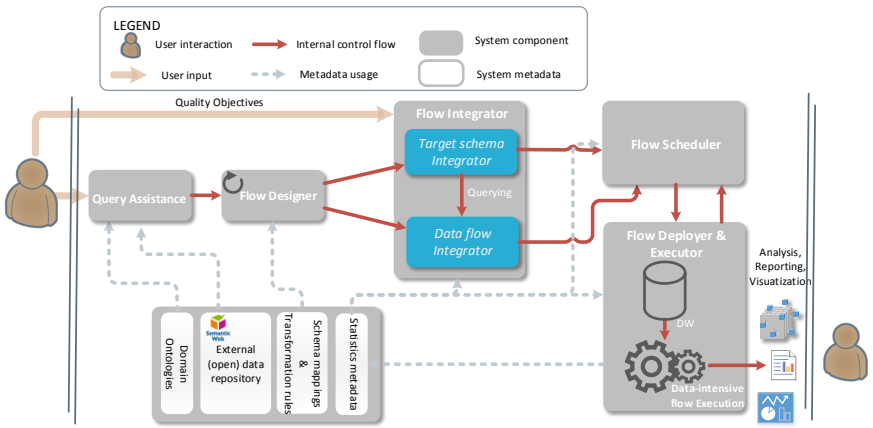

Fig. 6. Architecture for managing the lifecycle of data-intensive flows

new setting, and to this end, we envision here an architecture of a system for managing data-intensive flows (Figure 6).

The proposed architecture depicts at high level the main outcome of our study. It points out the main processing steps which need to be considered during the lifecycle of a data-intensive flow. Moreover, the architecture captures in a holistic way the complete lifecycle of data-intensive flows, and as such, it can be seen as a roadmap for both academia and industry toward building data-intensive flows in *next generation BI* systems. In what follows, we discuss in more detail different architectural modules, and if available, we point out example approaches that tackle the challenges of such modules.

We start with the *Query Assistance* module that should provide an intuitive interface to end users when expressing their information requirements. On the one hand, it should raise the usability of the BI system, for a broader set of business users. This module should provide a business-oriented view over the included data sources (e.g., by means of domain ontologies like in [79,50,5]). On the other hand, the *Query Assistance* module should also facilitate the low *coupledness* of data sources and be able to efficiently connect to a plethora of external data source repositories. These, preferably semantically enriched data sources (e.g., linked (open) data; [7]), should supplement user analysis with context-aware data and thus raise the *openness* of the delivered results (see Section 7).

Previously expressed information requirements further need to be automatically translated in an appropriate data flow (i.e., the *Flow Designer* module) that will satisfy such information requirements. *Flow Designer* must provide robustness for such loosely coupled systems in dealing with data sources with low (non-guaranteed) *accessibility*. Furthermore, to support the high *automation* of the *Flow Designer* module, the system should first revisit the existing approaches

for automatic schema mapping creation (e.g., [24]). Obviously, these approaches must be extended with more complex data transformations. First, supporting low *structuredness* and extracting useful data from unstructured data sources on-the-fly should be largely supported, as dynamic systems cannot always rely on having a batched preprocessing step doing so. Furthermore, more complex data analysis (e.g., machine learning algorithms) should be supported in these data flows. Here, we can benefit from exploiting different data and flow characteristics (e.g., by revisiting the previously studied field of intelligent assistants for data analysis [81]). Lastly, the *Flow Designer* module should automatically accommodate the flow to ensure the required level of data quality and output data *constraintness* (typically in contradiction with required data latency) [91]. Importantly, such a design process must be iterative to support high *malleability* of the data flow design in front of new, changed, or removed data sources or information requirements.

In the case of partially materializing data, as suggested in [15], the *Flow Designer* module should be aware or be able to reconstruct the target schema, where data are loaded in a previously executed batch process, and further queried when *interactively* answering information requirements. Thus, the final data flow ready for deployment must be integrated from the combination of data flows that directly access data sources and querying previously materialized target data (i.e., the *Flow Integrator* module). Notice that finding the optimal level of partial materialization is still a challenge and must be decided using previously collected flow statistics and following desired quality objectives.

Next, the optimization of data-intensive flows should be efficiently supported at different levels of the flow lifecyle (see Section 8. Initially, when the integrated data flow is created to answer a user's requirement at hand, optimization of a data flow should be done in combination with selecting the optimal partial materialization of data [74]. Furthermore, having multiple data-intensive flows answering different requirements of end-users waiting for execution, the system requires an optimal schedule for running these data flows over the shared computational resources (e.g., shared, multi-tenant cluster), i.e., *Flow Scheduler* module. Lastly, the automatic optimization means must be also provided when deploying data flows, for selecting an optimal execution engine (e.g., [85,56]), as well as for providing the lower level, engine-specific, optimization of a data flow (i.e., the *Flow Deployer* module).

From Figure 6 we can observe that the *automation* of the design and optimization of data-intensive flows, as well as the query assistance, must be largely facilitated by means of different metadata artifacts (i.e., schema mappings, domain ontology, flow and statistics). Indeed, the use of metadata for automating the design of the next generation data warehouse systems (DW 2.0) has been previously discussed [47], while recently the main challenges of matadata in the analytical process of the *next generation BI* systems have been studied in [95].

Finally, as an important remark, we want to draw a parallelism of the envisioned architecture depicted in Figure 6, and the traditional architecture of centralized database management systems (DBMS). First, using declarative (SQL)

queries in a DBMS, end users pose their analytical needs to the system. While based on the traditional database theory, the *semantic optimizer* is responsible for transforming a user query into an equivalent one with a lower cost, in *next generation BI* systems, user queries need to be additionally transformed and enriched to access external data by means of data exploration processes (i.e., *Query Assistance*; [2]). Furthermore, similarly to the *syntactic optimizer* in the traditional DBMSs, *Flow Designer* needs to translate an information requirement to a sequence of operations (i.e., *syntactic tree*), which represents a logical plan of a data-intensive flow execution. The execution plan should be typically optimized for an individual execution. However, in *next generation BI* systems, a data-intensive flow could also be integrated with other data flows for an optimized *multi-flow* execution (i.e., *Flow Integrator*). This conceptually resembles the well-known problem of multi-query optimization [80], but inevitably brings new challenges considering the complexity of data flow operations, which cannot always be presented using algebraic specifications [44]. Moreover, the *Flow Integrator* module should also transform input execution plan and optimize it considering partial materialization of data, similarly to the *query rewriting* techniques for answering queries using materialized views [42]. Following the traditional DBMS architecture, an integrated execution plan is then optimally scheduled for execution, together with the rest of the data flows in the system (i.e., *Flow Scheduler*). Lastly, the logical data flow is translated into the code of a selected execution engine (e.g., [51,56]), *physically optimized* considering available access structures, and finally deployed for *execution* (i.e., *Flow Deployer & Executor*). Similarly to the concept of the database catalog, throughout the lifecycle of a data-intensive flow, different metadata artifacts need to be available (e.g., schema mappings, transformation rules, statistics metadata; see Figure 6) to lead the automatic design and optimization of a data flow.

The parallelism drawn above finally confirms us that the underpinnings of data-intensive flows in *next generation BI* systems should be analyzed in the frame of the traditional DB theory field. Nevertheless, as we showed through our study, the inherent complexity of today's business environments (e.g., data heterogeneity, high complexity of data processing) must be additionally addressed, and comprehensively tackled to provide end-to-end solutions for managing the complete lifecycle of data-intensive flows.

10 Conclusions

In this paper, we studied data-intensive flows, focusing on the challenges of the next-generation BI setting. We analyzed the foundational work of database theory tackling heterogeneity and interoperability (i.e., data exchange and data integration), as well as the recent approaches both in the context of DW and *next generation BI* systems.

We first identified the main characteristics of data-intensive flows, which built the dimensions of our study setting, and further studied the current approaches

in the frame of these dimensions and determined the level the studied approaches attain in each of them.

As the main outcome of this study, we outline an architecture for managing the complexity of data-intensive flows in the *next generation BI* setting. We discuss in particular different components that such an architecture should realize, as well as the processes that the data-intensive flow lifecycle should carry out.

Finally, within the components of the envisioned architecture, we point out the main remaining challenges that the *next generation BI* setting brings to managing data-intensive flows, and which require special attention from both academia and industry.

Acknowledgements. This work has been partially supported by the Secreteria d'Universitats i Recerca de la Generalitat de Catalunya under 2014 SGR 1534, and by the Spanish Ministry of Education grant FPU12/04915.

References

1. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J.N., Naumann, F., Pedersen, T.B., Rizzi, S., Trujillo, J., Vassiliadis, P., Vossen, G.: Fusion Cubes: Towards Self-Service Business Intelligence. IJDWM 9(2), 66–88 (2013)
2. Abelló, A., Romero, O., Pedersen, T.B., Llavori, R.B., Nebot, V., Cabo, M.J.A., Simitsis, A.: Using semantic web technologies for exploratory OLAP: A survey. IEEE Trans. Knowl. Data Eng. 27(2), 571–588 (2015)
3. Akkaoui, Z.E., Mazón, J.N., Vaisman, A.A., Zimányi, E.: BPMN-Based Conceptual Modeling of ETL Processes. In: DaWaK. pp. 1–14 (2012)
4. Akkaoui, Z.E., Zimányi, E., Mazón, J.N., Trujillo, J.: A BPMN-Based Design and Maintenance Framework for ETL Processes. IJDWM 9(3), 46–72 (2013)
5. Bellatreche, L., Khouri, S., Berkani, N.: Semantic Data Warehouse Design: From ETL to Deployment à la Carte. In: DASFAA (2). pp. 64–83 (2013)
6. Berthold, H., Rösch, P., Zöller, S., Wortmann, F., Carenini, A., Campbell, S., Bisson, P., Strohmaier, F.: An architecture for ad-hoc and collaborative business intelligence. In: EDBT/ICDT Workshops (2010)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
8. Bruckner, R.M., List, B., Schiefer, J.: Striving towards near real-time data integration for data warehouses. In: Data Warehousing and Knowledge Discovery, 4th International Conference, DaWaK 2002, Aix-en-Provence, France, September 4-6, 2002, Proceedings. pp. 317–326 (2002)
9. Bruno, N., Jain, S., Zhou, J.: Continuous cloud-scale query optimization and processing. PVLDB 6(11), 961–972 (2013)
10. Buneman, P., Davidson, S.B., Fernandez, M.F., Suciu, D.: Adding Structure to Unstructured Data. In: ICDT. pp. 336–350 (1997)
11. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Data integration under integrity constraints. Inf. Syst. 29(2), 147–163 (2004)
12. Calì, A., Lembo, D., Rosati, R.: Query rewriting and answering under constraints in data integration systems. In: IJCAI. pp. 16–21 (2003)
13. Chen, C.L.P., Zhang, C.: Data-intensive applications, challenges, techniques and technologies: A survey on big data. Inf. Sci. 275, 314–347 (2014)

14. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44(3), 15 (2012)
15. Dayal, U., Castellanos, M., Simitsis, A., Wilkinson, K.: Data integration flows for business intelligence. In: EDBT. pp. 1–11 (2009)
16. Dayal, U., Kuno, H.A., Wiener, J.L., Wilkinson, K., Ganapathi, A., Krompass, S.: Managing operational business intelligence workloads. Operating Systems Review 43(1), 92–98 (2009)
17. Dayal, U., Wilkinson, K., Simitsis, A., Castellanos, M., Paz, L.: Optimization of Analytic Data Flows for Next Generation Business Intelligence Applications. In: TPCTC. pp. 46–66 (2011)
18. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
19. Dessloch, S., Hernández, M.A., Wisnesky, R., Radwan, A., Zhou, J.: Orchid: Integrating Schema Mapping and ETL. In: ICDE. pp. 1307–1316. IEEE (2008)
20. Doan, A., Halevy, A.Y., Ives, Z.G.: Principles of Data Integration. Morgan Kaufmann (2012)
21. Dong, X.L., Halevy, A.Y., Yu, C.: Data integration with uncertainty. VLDB J. 18(2), 469–500 (2009)
22. Eckerson, W.W.: Best practices in operational BI. Business Intelligence Journal 12(3), 7–9 (2007)
23. European Commission: G. technology readiness levels (TRL) (2014)
24. Fagin, R., Haas, L.M., Hernández, M.A., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange. In: Conceptual Modeling: Foundations and Applications. pp. 198–236. Springer (2009)
25. Fagin, R., Kimelfeld, B., Kolaitis, P.G.: Probabilistic data exchange. Journal of the ACM (JACM) 58(4), 15 (2011)
26. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data Exchange: Semantics and Query Answering. In: ICDT. pp. 207–224. Springer (2003)
27. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
28. Feldman, R., Sanger, J.: The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press (2007)
29. Ferrara, E., Meo, P.D., Fiumara, G., Baumgartner, R.: Web data extraction, applications and techniques: A survey. Knowl.-Based Syst. 70, 301–323 (2014)
30. Friedman, M., Levy, A.Y., Millstein, T.D.: Navigational Plans for Data Integration. In: Intelligent Information Integration (1999)
31. García, S., Romero, O., Ravents, R.: DSS from an RE perspective: A systematic mapping. Journal of Systems and Software 117, 488 – 507 (2016)
32. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The TSIMMIS Approach to Mediation: Data Models and Languages. J. Intell. Inf. Syst. 8(2), 117–132 (1997)
33. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., Jacobsen, H.A.: BigBench: towards an industry standard benchmark for big data analytics. In: SIGMOD Conference. pp. 1197–1208 (2013)
34. Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: On reconciling data exchange, data integration, and peer data management. In: PODS. pp. 133–142 (2007)
35. Giannikis, G., Makreshanski, D., Alonso, G., Kossmann, D.: Shared workload optimization. PVLDB 7(6), 429–440 (2014)
36. Giorgini, P., Rizzi, S., Garzetti, M.: Grand: A goal-oriented approach to requirement analysis in data warehouses. DSS 45(1), 4–21 (2008)

37. Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. Data Knowl. Eng. 59(2), 435–459 (2006)
38. Golfarelli, M., Rizzi, S.: Data Warehouse Design. Modern Principles and Methodologies. McGraw-Hill (2009)
39. Golfarelli, M., Rizzi, S., Cella, I.: Beyond data warehousing: what's next in business intelligence? In: DOLAP. pp. 1–6 (2004)
40. Haas, L.M.: Beauty and the Beast: The Theory and Practice of Information Integration. In: ICDT. pp. 28–43 (2007)
41. Halasipuram, R., Deshpande, P.M., Padmanabhan, S.: Determining essential statistics for cost based optimization of an ETL workflow. In: EDBT. pp. 307–318 (2014)
42. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. 10(4), 270–294 (2001)
43. Heimbigner, D., McLeod, D.: A Federated Architecture for Information Management. ACM Trans. Inf. Syst. 3(3), 253–278 (1985)
44. Hueske, F., Peters, M., Sax, M., Rheinländer, A., Bergmann, R., Krettek, A., Tzoumas, K.: Opening the Black Boxes in Data Flow Optimization. PVLDB 5(11), 1256–1267 (2012)
45. IBM, Zikopoulos, P., Eaton, C.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media, 1st edn. (2011)
46. Inmon, W.H.: Building the Data Warehouse. John Wiley & Sons, Inc. (1992)
47. Inmon, W.H., Strauss, D., Neushloss, G.: DW 2.0: The architecture for the next generation of data warehousing: The architecture for the next generation of data warehousing. Morgan Kaufmann (2010)
48. Jarke, M., Koch, J.: Query Optimization in Database Systems. ACM Comput. Surv. 16(2), 111–152 (1984)
49. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Incremental consolidation of data-intensive multi-flows. IEEE Trans. Knowl. Data Eng. 28(5), 1203–1216 (2016)
50. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A., Candón, H., Nadal, S.: Quarry: Digging up the gems of your data treasury. In: EDBT. pp. 549–552 (2015)
51. Jovanovic, P., Simitsis, A., Wilkinson, K.: Engine independence for logical analytic flows. In: ICDE. pp. 1060–1071 (2014)
52. Karagiannis, A., Vassiliadis, P., Simitsis, A.: Scheduling strategies for efficient ETL execution. Inf. Syst. 38(6), 927–945 (2013)
53. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit. John Wiley & Sons (2004)
54. Kirk, T., Levy, A.Y., Sagiv, Y., Srivastava, D., Others: The information manifold. In: Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments. vol. 7, pp. 85–91 (1995)
55. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: PODS. pp. 61–75 (2005)
56. Kougka, G., Gounaris, A., Tsichlas, K.: Practical algorithms for execution engine selection in data flows. Future Generation Computer Systems 45, 133–148 (2015)
57. Labio, W., Garcia-Molina, H.: Efficient Snapshot Differential Algorithms for Data Warehousing. In: VLDB. pp. 63–74 (1996)
58. Laender, A.H.F., Ribeiro-Neto, B.A., da Silva, A.S., Teixeira, J.S.: A Brief Survey of Web Data Extraction Tools. SIGMOD Record 31(2), 84–93 (2002)
59. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: PODS. pp. 233–246. ACM (2002)

60. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press (2014)
61. Libkin, L.: Data exchange and incomplete information. In: PODS. pp. 60–69 (2006)
62. Lindsay, B.G., Haas, L.M., Mohan, C., Pirahesh, H., Wilms, P.F.: A Snapshot Differential Refresh Algorithm. In: SIGMOD Conference. pp. 53–60 (1986)
63. Löser, A., Hueske, F., Markl, V.: Situational Business Intelligence. In: BIRTE. vol. 27, pp. 1–11 (2008)
64. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modeling. Data Knowl. Eng. 68(12), 1452–1469 (2009)
65. McBrien, P., Poulovassilis, A.: Data Integration by Bi-Directional Schema Transformation Rules. In: ICDE. pp. 227–238 (2003)
66. McDonald, K., Wilmsmeier, A., Dixon, D.C., Inmon, W.: Mastering the SAP Business Information Warehouse. John Wiley & Sons (2002)
67. Morzy, T., Wrembel, R.: On querying versions of multiversion data warehouse. In: DOLAP. pp. 92–101 (2004)
68. Muñoz, L., Mazón, J.N., Trujillo, J.: Automatic generation of ETL processes from conceptual models. In: DOLAP. pp. 33–40 (2009)
69. Ong, K.W., Papakonstantinou, Y., Vernoux, R.: The SQL++ semi-structured data model and query language: A capabilities survey of sql-on-hadoop, nosql and newsql databases. CoRR abs/1405.3631 (2014), http://arxiv.org/abs/1405.3631
70. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: Policy-regulated management of ETL evolution. J. Data Semantics 13, 147–177 (2009)
71. Patel-Schneider, P.F., Horrocks, I.: Position paper: a comparison of two modelling paradigms in the Semantic Web. In: WWW. pp. 3–12 (2006)
72. Pohl, K.: Requirements Engineering - Fundamentals, Principles, and Techniques. Springer (2010)
73. Polo, J., Becerra, Y., Carrera, D., Torres, J., Ayguadé, E., Steinder, M.: Adaptive MapReduce scheduling in shared environments. In: IEEE/ACM CCGrid. pp. 61–70 (2014)
74. Qu, W., Dessloch, S.: A real-time materialized view approach for analytic flows in hybrid cloud environments. Datenbank-Spektrum 14(2), 97–106 (2014)
75. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Eng. Bull. 23(4), 3–13 (2000)
76. Rheinlnder, A., Heise, A., Hueske, F., Leser, U., Naumann, F.: Sofa: An extensible logical optimizer for UDF-heavy data flows. Information Systems 52(0), 96 – 125 (2015)
77. Romero, O., Abelló, A.: A Survey of Multidimensional Modeling Methodologies. IJDWM 5(2), 1–23 (2009)
78. Romero, O., Abelló, A.: Open Access Semantic Aware Business Intelligence. In: Business Intelligence, Lecture Notes in Business Information Processing, vol. 172, pp. 121–149. Springer (2014)
79. Romero, O., Simitsis, A., Abelló, A.: GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In: DaWaK. vol. 6862, pp. 80–95. Springer (2011)
80. Roy, P., Sudarshan, S.: Multi-query optimization. In: Encyclopedia of Database Systems, pp. 1849–1852. Springer US (2009)
81. Serban, F., Vanschoren, J., Kietz, J., Bernstein, A.: A survey of intelligent assistants for data analysis. ACM Comput. Surv. 45(3), 31 (2013)

82. Shu, N.C., Housel, B.C., Taylor, R.W., Ghosh, S.P., Lum, V.Y.: EXPRESS: A Data EXtraction, Processing, amd REStructuring System. ACM Trans. Database Syst. 2(2), 134–174 (1977)
83. Simitsis, A., Vassiliadis, P., Sellis, T.K.: State-Space Optimization of ETL Workflows. IEEE Trans. Knowl. Data Eng. 17(10), 1404–1419 (2005)
84. Simitsis, A., Wilkinson, K., Castellanos, M., Dayal, U.: QoX-driven ETL design: reducing the cost of ETL consulting engagements. In: SIGMOD. pp. 953–960 (2009)
85. Simitsis, A., Wilkinson, K., Castellanos, M., Dayal, U.: Optimizing analytic data flows for multiple execution engines. In: SIGMOD Conference. pp. 829–840 (2012)
86. Simitsis, A., Wilkinson, K., Dayal, U., Hsu, M.: HFMS: managing the lifecycle and complexity of hybrid analytic data flows. In: ICDE. pp. 1174–1185 (2013)
87. Skoutas, D., Simitsis, A.: Designing ETL processes using semantic web technologies. In: DOLAP. pp. 67–74 (2006)
88. Skoutas, D., Simitsis, A.: Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data. Int. J. Semantic Web Inf. Syst. 3(4), 1–24 (2007)
89. Strange, K.H.: ETL Was the Key to This Data Warehouse's Success. Gartner Research, CS-15-3143 (2002)
90. Theodoratos, D., Sellis, T.K.: Designing Data Warehouses. Data Knowl. Eng. 31(3), 279–301 (1999)
91. Theodorou, V., Abelló, A., Thiele, M., Lehner, W.: POIESIS: a tool for quality-aware ETL process redesign. In: EDBT. pp. 545–548 (2015)
92. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: ER. pp. 307–320 (2003)
93. Tziovara, V., Vassiliadis, P., Simitsis, A.: Deciding the physical implementation of ETL workflows. In: DOLAP. pp. 49–56 (2007)
94. Ullman, J.D.: Information Integration Using Logical Views. In: ICDT. pp. 19–40 (1997)
95. Varga, J., Romero, O., Pedersen, T.B., Thomsen, C.: Towards next generation BI systems: The analytical metadata challenge. In: DaWaK. pp. 89–101 (2014)
96. Vassiliadis, P.: A survey of extract-transform-load technology. IJDWM 5(3), 1–27 (2009)
97. Vassiliadis, P., Simitsis, A.: Near real time ETL. In: New Trends in Data Warehousing and Data Analysis, pp. 1–31. Springer (2009)
98. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Inf. Syst. 30(7), 492–525 (2005)
99. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: DOLAP. pp. 14–21 (2002)
100. Waas, F., Wrembel, R., Freudenreich, T., Thiele, M., Koncilia, C., Furtado, P.: On-demand ELT architecture for right-time BI: extending the vision. IJDWM 9(2), 21–38 (2013)
101. Wilkinson, K., Simitsis, A., Castellanos, M., Dayal, U.: Leveraging business process models for ETL design. In: ER. pp. 15–30 (2010)
102. Winter, R., Strauch, B.: A Method for Demand-driven Information Requirements Analysis in Data Warehousing Projects. In: In Proc. HICSS. pp. 1359–1365 (2003)